The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Trinocular Ground System to Control UAVs

Carol Martínez, Pascual Campoy, Iván Mondragón, and Miguel A. Olivares-Méndez
Computer Vision Group
Universidad Politécnica de Madrid
Jose Gutierrez Abascal 2, 28006 Madrid, Spain
carolviviana.martinez@upm.es
www.disam.upm.es/colibri

Abstract— **In this paper we introduce a real-time trinocular system to control rotary wing Unmanned Aerial Vehicles based on the 3D information extracted by cameras located on the ground. The algorithm is based on key features onboard the UAV to estimate the vehicle's position and orientation. The algorithm is validated against onboard sensors and known 3D positions, showing that the proposed camera configuration robustly estimates the helicopter's position with an adequate resolution, improving the position estimation, especially the height estimation. The obtained results show that the proposed algorithm is suitable to complement or replace the GPS-based position estimation in situations where GPS information is unavailable or where its information is inaccurate, allowing the vehicle to develop tasks at low heights, such as autonomous landing, take-off, and positioning, using the extracted 3D information as a visual feedback to the flight controller.**

I. INTRODUCTION

The use of computer vision algorithms designed to control UAVs has been an attractive and interesting research area in the academic and industrial field, being a special case of study the use of autonomous helicopters due to their grate maneuver capabilities that allow them to fly at low speed velocities (lateral and longitudinal flights), hover, perform vertical take-off and landing, and maneuver in reduced spaces.

The vision systems implemented in UAVs cover areas such as object detection and object tracking [1], pose estimation [2], navigation [3], obstacle detection [4], and autonomous landing [5], among others, in which vision is always used as a rich source of information for the mission's success.

The goal of this research is to provide UAVs with an additional vision-based source of information, extracted by ground cameras, in order to allow UAVs to develop visually guided tasks, such as landing, inspection or ground-air cooperation missions, especially in situations when the GPS information is not available, when the GPS-based position estimation is not accurate enough for the task to develop, or when the payload restrictions do not allow the incorporation of additional sensors.

The vision based pose estimation problem has been extensively studied, such as in [6], [7], [8], and [9], where the information of the vision system is integrated with the onboard sensors to control the UAV. In another approach, an onboard camera, in conjunction with a ground moire pattern, is used to estimate the vehicle's six degrees of freedom for landing purposes [10]. Ground-based cameras approaches for solving the pose estimation problem have also been proposed. In [11], a ground camera is used to control a *dirigible* based on the information of artificial landmarks. In that work, the 3D position and orientation information are recovered based on the knowledge of the geometrical properties of the landmarks. A more recent work deals with the pose estimation problem of a *quadrotor* vehicle using simultaneously both ground and onboard cameras [12]. In the latter work, color landmarks are used to recover the vehicle's pose. In this paper, we introduce a vision-based control system for UAVs by presenting the estimation of the position and orientation (*Yaw* angle) of a rotary wing Unmanned Aerial Vehicle and its integration in the UAV control loop. All of this, has the aim of developing a vision-based platform for autonomous take-off and landing, and high precision positioning tasks. The implemented system is a redundant system composed of three cameras located on the ground in charge of recovering the information of key features onboard the helicopter in order to obtain a robust 3D position estimation by using a trinocular or binocular estimation (depending on the number of cameras that see a specific key feature), and also to increase the vehicle's workspace.

In these first steps towards an external vision-based control, color landmarks onboard the UAV have been used as key features. The helicopter detection and tracking is based on such landmarks, and is achieved by using the CamShift (Continuously Adaptive Mean SHIFT) algorithm [13]. 3D reconstruction techniques are also used for the estimation of the helicopter's pose [14].

The proposed algorithm has been tested in a variety of situations (indoors and outdoors), and the results of these tests are shown in this paper, whose outline is as follows: Section II presents the system architecture. Section III describes the reference systems. Section IV deals with the vision-based control system for UAVs. Finally, in Sections V and VI experiment results, conclusions and future works are presented.

II. SYSTEM ARCHITECTURE

The adequate intervention of several components from a hardware and software perspective is required for a mission's success, especially when a vision system is used in a UAV

control loop. The following section presents a description of the different components of the proposed system.

Fig. 1. Helicopter tesbed Colibri III, during a flight. Color landmarks onboard the UAV are used as key features for the position estimation algorithm.

A. Helicopter Testbed

The Computer Vision Group at the UPM (Universidad Politcnica de Madrid) counts with three fully operational UAV systems to develop indoors and outdoors applications. Such systems form part of the COLIBRI Project, whose purpose is to develop algorithms for vision based control of UAVs [15]. The experiments were carried out with the Colibri III system (see Fig. 1). It is a Rotomotion SR20 helicopter with an electric motor of 1.300 W, 8A. It is equipped with an xscale-based flight computer, a GPS, an IMU (Inertial Measurement Unit), an onboard pan and tilt camera, and an onboard Nano Itx computer for image processing algorithms. The flight computer runs Linux OS and uses an 802.11g wireless Ethernet protocol for sending and receiving information to/from the ground station.

B. Trinocular System

The trinocular system, as shown in Fig. 2, is composed of three FireWire cameras with 3.4mm lenses (Field Of View: $H_{FOV} \approx 87^\circ$ $V_{FOV} \approx 71^\circ$); each camera captures images of 320×240 size at 30fps (frames per second). These cameras are strategically located on an aluminum platform in order to allow a convergent optical axis configuration in two of the three cameras, and to take advantage of the cameras' field of view. The cameras' position and orientation are fixed, and their intrinsic and extrinsic parameters are known due to a calibration process. The cameras are connected to a vision computer, which is a 2.2 GHz laptop running Linux OS.

C. Communication System

We use a client-server architecture implemented in our systems (Colibri I, Colibri II, and Colibri III). This architecture is based on TCP/UDP messages and is used to exchange information between the helicopter, the ground station and the ground vision system. The exchange is achieved through a high level layer defined by a communication API (Application Programming Interface), which routes the messages to the specific process depending on the message type. The messages can be sent from the ground station or from the vision system, to the flight controller and can also be sent in the other direction (from the flight controller to the ground station and the vision system). Those messages include velocity control, position control, and helicopter and GPS state messages, among others. Fig. 3 shows an example of messages that can be transmitted through the switching layer.

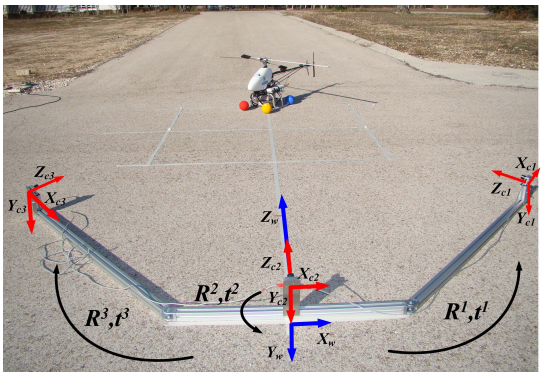

Fig. 2. Trinocular system. It is composed by three FireWire cameras that capture images of 320×240 size at 30fps. The rotation matrices and translation vectors that define the relation of the cameras with the *World Coordinate System* are calculated from a calibration process.

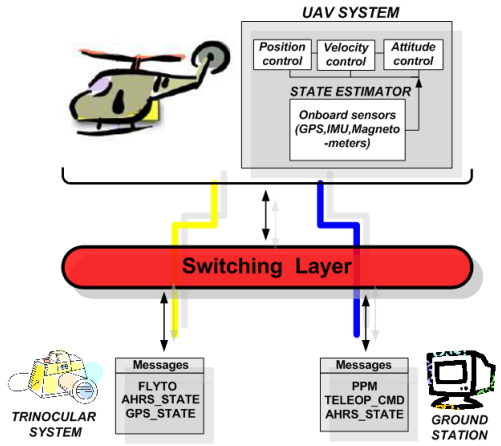

Fig. 3. Switching layer. TCP/UDP messages are used to exchange data between the flight controller, the ground station, and the vision system.

III. COORDINATE SYSTEMS

Different coordinate systems are used to map the extracted visual information from $\Re^2$ to $\Re^3$, and then to convert this information into commands to the helicopter. This section provides a description of the coordinate systems and their corresponding transformations to achieve vision-based tasks.

We have different coordinate systems; the *Image Coordinate System* (X_i), that includes the *Lateral* (X_f) and *Central Coordinate Systems* (X_u) in the image plane, the *Camera Coordinate System* (X_c), the *Helicopter Coordinate System* (X_h), and an additional one: the *World Coordinate System* (X_w), used as the principal reference system to control the vehicle (see Fig. 4).

- ***Image* and *Camera Coordinate Systems***

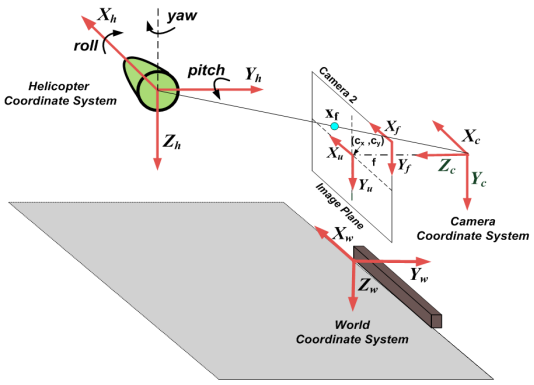

Fig. 4. Local and global coordinate systems. The mapping from the image plane to the *Camera Coordinate System* requires the calculation of the intrinsic camera parameters (c_x,c_y,f). Transformation from the *Camera Coordinate System* to *World Coordinate System* is performed through a rigid transformation, and requires the extrinsic parameters of the trinocular system. The *World* and the *Helicopter Coordinate Systems* are related through a rigid transformation, which defines the helicopter's position and orientation, and allows to send the adequate commands to the flight controller.

The relation between the *Camera Coordinate System* and the *Image Coordinate System* is taken from the "*pinhole*" camera model. It states that any point referenced in the *Camera Coordinate System* $\mathbf{x_c}$ is projected onto the image plane in the point $\mathbf{x_f}$ by intersecting the ray that links the 3D point $\mathbf{x_c}$ with the center of projection and the image plane. This mapping is described in (1), where $\mathbf{x_c}$ and $\mathbf{x_f}$ are represented in homogenous coordinates.

$$\begin{bmatrix} nx_f \\ ny_f \\ n \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \qquad (1)$$

$$\mathbf{x_f} = \mathbf{K^k}[\mathbf{I}|\mathbf{0}]\mathbf{x_c}$$

The matrix $\mathbf{K^k}$ contains the intrinsic camera parameters of the k^{th} camera, such as the coordinates of the center of projection (c_x, c_y) in pixel units, and the focal length (f_x, f_y), where $f_x = fm_x$ and $f_y = fm_y$ represent the focal length in terms of pixel dimensions, being m_x and m_y the number of pixels per unit distance.

The above-mentioned camera model assumes that the world point, the image point, and the optical center are collinear; however, in a real camera lens there are some effects (lens distortions) that have to be compensated in order to have a complete model. This compensation can be achieved by the calculation of the distortion coefficients through a calibration process [16], in which the intrinsic camera parameters, as well as the radial and tangential distortion coefficients, are calculated.

- ***Camera* and *World Coordinate Systems***

Considering that the cameras are fixed, these systems are related by a rigid transformation that allows to define the pose of the k^{th} camera in a *World Coordinate Frame*. As presented in (2), this transformation is defined by a rotation matrix $\mathbf{R^k}$ and a translation vector $\mathbf{t^k}$ that link the two coordinate systems and represent the extrinsic camera parameters. Such parameters are calculated through a calibration process of the trinocular system.

$$\mathbf{x_c} = \begin{bmatrix} \mathbf{R^k} & \mathbf{t^k} \\ \mathbf{0^T} & 1 \end{bmatrix} \mathbf{x_w} \qquad (2)$$

- ***World* and *Helicopter Coordinate Systems***

The *Helicopter Reference System*, as described in Fig. 4, has its origin at the center of mass of the vehicle and its correspondent axes: X_h, aligned with the helicopter's longitudinal axis; Y_h, transversal to the helicopter; and Z_h, pointing down. Considering that the estimation of the helicopter's pose with respect to the *World Coordinate System* is based on the distribution of the landmarks around the *Helicopter Coordinate System*, and that the information extracted from the vision system will be used as reference to the flight controller, a relation between those coordinate systems has to be found.

In Fig. 4, it is possible to observe that this relation depends on a translation vector that defines the helicopter's position ($\mathbf{t}$), and on a rotation matrix $\mathbf{R}$ that defines the orientation of the helicopter (*pitch*, *roll* and *yaw* angles). For practical purposes, *pitch* and *roll* angles are considered $= 0$ (helicopter flying at low velocities ($< 4m/s$), and only the *yaw* angle (θ) is taken into account in order to send the adequate commands to the helicopter. Therefore, the relation of the *World* and the *Helicopter Coordinate Systems* can be expressed as follows:

$$\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 & t_x \\ sin(\theta) & cos(\theta) & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_h \\ y_h \\ z_h \\ 1 \end{bmatrix} \qquad (3)$$

Where (t_x, t_y, t_z) will represent the position of the helicopter ($x_{w_{uav}}$,$y_{w_{uav}}$,$z_{w_{uav}}$) with respect to the *World Coordinate System*, and θ the helicopter's orientation.

IV. VISION-BASED CONTROL

A. Feature extraction and tracking

A color-based algorithm is applied to extract four different color landmarks (i), which are located onboard the UAV as shown in Fig. 1. The algorithm that is implemented for color segmentation is based on probability distributions. As a consequence, a model probability distribution of the color of each landmark and a probability distribution of those colors in each frame have to be determined. Converting from *RGB* to *HSV* color spaces, and using the histogram of the *Hue* channel, the color probability distributions are created. Then, the relation between the probability distributions (the models' ones with the probability distribution of each color in each frame) is found using the backprojection algorithm proposed by *Swain* and *Ballard* in [17]. This algorithm computes the likelihood that each image point belongs to the model. This relation is found by calculating a ratio histogram *Rh*, as defined in (4).

$$Rh_i^k(j) = min\left[\frac{Mh_i^k(j)}{Ih_i^k(j)}, 1\right] \qquad (4)$$

Where $Mh_i^k(j)$, $Ih_i^k(j)$, and $Rh_i^k(j)$ correspond respectively to the values of the j^{th} bin of the model histogram, the current image histogram, and the ratio histogram created for the i^{th} color in the k^{th} camera. Once Rh_i^k is found, it is then backprojected onto the image, replacing the values of the image by the values of Rh_i^k that the former values index. The resulting image is a gray-scale image, where pixels' values represent the probability that the pixel belongs to the object of interest. Once each backprojected image has been extracted, a threshold is applied in order to eliminate outliers.

The new location of the object in each frame is found by using the backprojected image previously extracted and the *Continuously Adaptive Mean Shift* (*CamShift*) algorithm [13] to track the landmarks. This tracker is based on the $Mean-Shift$ algorithm, originally introduced in [18]. This algorithm operates on probability distributions [19] and uses the gradient of the distributions in order to find the nearest dominant mode (peak). However, taking into account that this distribution can change over time, the *CamShift* algorithm allows the $Mean-Shift$ algorithm to adapt dynamically to those changes based on the zeroth moment information of the search window [13].

The following steps summarize the *CamShift* algorithm [20]:

1) Choose the initial location and size of the search window.
2) Run $Mean-Shift$ algorithm (one or many iterations) to find the densest region. Once it is found, store the zeroth moment and its central location.
3) Center the search window at the mean location found in Step 2 and set the search window size equal to a function of the zeroth moment found in Step 2.
4) Repeat Steps 2 to 3 until converge occurs (when the mean location moves less than a preset threshold).

For each frame in each camera, the *CamShift* algorithm finds the centroid of the landmark $(\bar{x}_i^k, \bar{y}_i^k)$ within the search window by calculating the zeroth $(m_{i_{00}}^k)$ and first moments in the X and Y axes $(m_{i_{10}}^k,\ m_{i_{01}}^k)$, as presented in (5).

$$\begin{gathered} m_{i_{00}}^k = \Sigma_x \Sigma_y I_i^k(x,y) \\ m_{i_{10}}^k = \Sigma_x \Sigma_y x I_i^k(x,y),\ \ m_{i_{01}}^k = \Sigma_x \Sigma_y y I_i^k(x,y) \\ \bar{x}_i^k = \frac{m_{i_{10}}^k}{m_{i_{00}}^k};\ \ \bar{y}_i^k = \frac{m_{i_{01}}^k}{m_{i_{00}}^k} \end{gathered} \tag{5}$$

Where $I_i^k(x,y)$ represents the intensity value (within the search window) of the backprojected image found for landmark i in the k^{th} camera. The centroid found is then used as feature for the 3D reconstruction stage.

B. 3D Reconstruction

The 3D reconstruction of each landmark requires a previous calibration of the system in order to find the intrinsic parameters ($\mathbf{K^k}$) of the cameras and the extrinsic parameters ($\mathbf{R^k}$ and $\mathbf{t^k}$) of the trinocular system (Fig. 2). This calibration is done using the *CalTech* camera calibration toolbox [21]. Once the previously mentioned parameters are calculated for each camera, the different transformations mentioned in Section III have to be applied from the image plane to the *World Coordinate System* in order to recover the 3D position of the landmarks.

Reorganizing (1), we have that:

$$x_{f_i} - c_x^k = f_x^k \frac{x_{c_i}}{z_{c_i}}, \quad y_{f_i} - c_y^k = f_y^k \frac{y_{c_i}}{z_{c_i}} \tag{6}$$

If we consider that $x_{u_i} = x_{f_i} - c_x^k$, $y_{u_i} = y_{f_i} - c_y^k$, and if we take into account the color information of the landmarks and the trinocular extrinsic parameters of the trinocular system, equation (2) can be applied to relate a 3D point (the 3D position of landmark i) with its projection in each camera using the following equation:

$$\begin{aligned} x_{u_i}^k &= f^k \frac{r_{11}^k x_{w_i} + r_{12}^k y_{w_i} + r_{13}^k z_{w_i} + t_x^k}{r_{31}^k x_{w_i} + r_{32}^k y_{w_i} + r_{33}^k z_{w_i} + t_z^k} \\ y_{u_i}^k &= f^k \frac{r_{21}^k x_{w_i} + r_{22}^k y_{w_i} + r_{23}^k z_{w_i} + t_y^k}{r_{31}^k x_{w_i} + r_{32}^k y_{w_i} + r_{33}^k z_{w_i} + t_z^k} \end{aligned} \tag{7}$$

In the aforementioned equations, $x_{u_i}^k$ and $y_{u_i}^k$ represent the coordinates of landmark i expressed in the Central *Camera Coordinate System* of the k^{th} camera, whereas, r^k and t^k are the components of the rotation matrix $\mathbf{R^K}$ and the translation vector $\mathbf{t^k}$ that represent the extrinsic parameters of the trinocular system.

Therefore, for each landmark in each camera we will obtain two equations such as those presented in (7). Applying a restriction regarding the minimum number of cameras that see a specific landmark (in order to triangulate the landmark position), if at least two cameras are seeing the same landmark, it is possible to create a system of equations of the form $\mathbf{Ac} = \mathbf{b}$,

where:

$$\mathbf{A} = \begin{bmatrix} (x_{c_i}^1 r_{31}^1 - r_{11}^1) & (x_{c_i}^1 r_{32}^1 - r_{12}^1) & (x_{c_i}^1 r_{33}^1 - r_{13}^1) \\ (y_{c_i}^1 r_{31}^1 - r_{21}^1) & (y_{c_i}^1 r_{32}^1 - r_{22}^1) & (y_{c_i}^1 r_{33}^1 - r_{23}^1) \\ \vdots & \vdots & \vdots \\ (x_{c_i}^k r_{31}^k - r_{11}^k) & (x_{c_i}^k r_{32}^k - r_{12}^k) & (x_{c_i}^k r_{33}^k - r_{13}^k) \\ (y_{c_i}^k r_{31}^k - r_{21}^k) & (y_{c_i}^k r_{32}^k - r_{22}^k) & (y_{c_i}^k r_{33}^k - r_{23}^k) \end{bmatrix} \tag{8}$$

$$\mathbf{c} = \begin{bmatrix} x_{w_i} \\ y_{w_i} \\ z_{w_i} \end{bmatrix} \tag{9}$$

$$\mathbf{b} = \begin{bmatrix} (t_x^1 - x_{c_i}^k t_z^1) \\ (t_y^1 - y_{c_i}^k t_z^1) \\ \vdots \\ (t_x^k - x_{c_i}^k t_z^k) \\ (t_y^k - y_{c_i}^k t_z^k) \end{bmatrix} \tag{10}$$

The previous system can be solved using the pseudo-inverse of the matrix $\mathbf{A}$, as presented in (11), where the obtained vector $\mathbf{c}$ represents the position $(x_{w_i}, y_{w_i}, z_{w_i})$ of the i^{th} landmark.

$$\mathbf{c} = \mathbf{A}^+\mathbf{b} = (\mathbf{A^T A})^{-1}\mathbf{A^T b} \tag{11}$$

C. Pose estimation

In this case, the estimation of the helicopter's pose will be related exclusively with the estimation of the 3D position of the helicopter ($\mathbf{x_{w_{uav}}}$) and the orientation of the helicopter both expressed with respect to the *World Coordinate System*. The helicopter's orientation is defined only with respect to the Z_h axis (θ), while the angles, with respect to the other axes, are considered $= 0$. Therefore, taking into account the 3D positions of the landmarks extracted in the previous stage, and taking into account the landmarks' distribution around the *Helicopter Coordinate System*, it is possible to formulate equation (12) for each landmark, considering only the *yaw* angle θ (as explained in section III):

$$\begin{bmatrix} x_{w_i} \\ y_{w_i} \\ z_{w_i} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & t_x \\ \sin(\theta) & \cos(\theta) & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{h_i} \\ y_{h_i} \\ z_{h_i} \\ 1 \end{bmatrix} \quad (12)$$

Reorganizing (12) for all the landmarks that have been detected, and considering that $\mathrm{c}\theta = \cos(\theta)$, $\mathrm{s}\theta = \sin(\theta)$, $x_{w_{uav}} = t_x$, $y_{w_{uav}} = t_y$, $z_{w_{uav}} = t_z$, equation (12) can be rewritten in the form:

$$\mathbf{Ac} = \mathbf{b} \quad (13)$$

$$\begin{bmatrix} x_{h_1} & y_{h_1} & 1 & 0 & 0 \\ y_{h_1} & -x_{h_1} & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{h_i} & y_{h_i} & 1 & 0 & 0 \\ y_{h_i} & -x_{h_i} & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathrm{c}\theta \\ \mathrm{s}\theta \\ x_{w_{uav}} \\ y_{w_{uav}} \\ z_{w_{uav}} \end{bmatrix} = \begin{bmatrix} x_{w_1} \\ y_{w_1} \\ z_{w_1} - z_{h_1} \\ \vdots \\ x_{w_i} \\ y_{w_i} \\ z_{w_i} - z_{h_i} \end{bmatrix}$$

Where our unknown parameters $\mathrm{c}\theta, \mathrm{s}\theta, x_{w_{uav}}, y_{w_{uav}}, z_{w_{uav}}$ correspond to the values that define the orientation (*yaw* angle) and the position of the helicopter with respect to the *World Coordinate System*. The previous system of equations can be solved as in (11), with the restriction that the reconstruction of at least two landmarks is required to find both $\mathbf{x_{w_{uav}}}$ and the *yaw* angle (θ). However, this restriction can vary depending on the task to develop: in the case of autonomous landing, if we assume that the helicopter is in the adequate landing position and the trinocular system will actuate only on the helicopter's height, the control task can be achieved with only one detected landmark.

D. Vision system and flight control system integration

The integration of the visual system into the UAV control system is presented in Fig. 5. It follows the architecture of a *"dynamic look and move system"* [22], which is a hierarchical control architecture composed of a fast internal control loop -flight control system- ($\pm 120Hz$), which is in charge of the helicopter's stability, and an external loop (the vision system), which operates at low frequencies and gives references to the helicopter's control loop.

The helicopter's control loop is composed of: a *State Estimator*, which fuses the information from different sensors (GPS, Magnetometers, IMU -Inertial Measurement Unit-) using a *Kalmanfilter* to determine the position and orientation of the vehicle, and the *Flight Controller* which is composed of different control loops based on PID controllers (for attitude control, velocity control, position control) that allow different modes of operation, as presented in [23], [15], and [24].

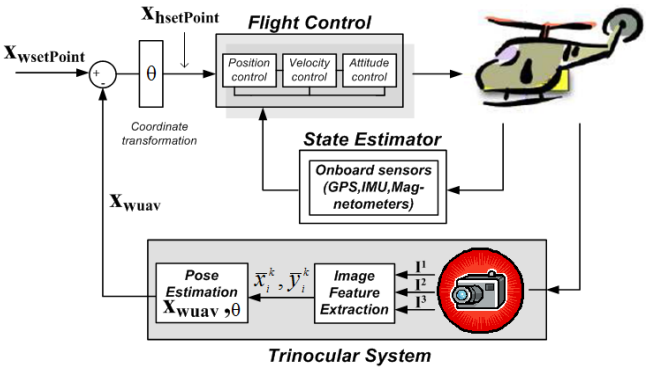

Fig. 5. Vision-based control. In this scheme, a fast internal control loop (flight control system) is in charge of the helicopter's stability, whereas the external loop (vision system) operates at low frequency and uses the visual information as a feedback to the helicopter's flight controller.

Considering that the vision system will determine the position of the UAV in the *World Coordinate System*, the adequate references will be given in terms of 3D positions, so that the position $\mathbf{x_{w_{setPoint}}}$ and the position information given by the trinocular system $\mathbf{x_{w_{uav}}}$, both defined in the *World Coordinate System*, will be compared to generate references to the position controller, as shown in Fig. 5. These references are first transformed into commands to the helicopter $\mathbf{x_{h_{setPoint}}}$ by taking into account the helicopter's orientation θ (see Fig. 5), and then those references are sent to the position controller in order to move the helicopter to the desired position.

V. EXPERIMENTS AND RESULTS

Several experiments were developed with the aim of validating the proposed algorithm regarding feature extraction, tracking and position estimation. The program was developed in $C++$, and the *OpenCV* libraries were used for image processing. The UAV system and the trinocular platform described in Section II were used for the tests.

- **Static tests**

The first set of experiments consisted in positioning the helicopter in different known 3D coordinates with respect to the *World Coordinate System*. Real values of the UAV positions are obtained measuring them. The Root Mean Square Error (RMSE) of the helicopter's position is calculated by comparing the algorithm results with the known position. Fig. 6 presents the RMSE for different situations. As expected, the error in the Y_w axis increases as the helicopter gets farther away from the trinocular platform. This is so because at those positions the parallax angle within the rays becomes small. Errors in Z_w and X_w axes are satisfactory and constant within a small range, compared with those obtained

in the Y_w direction. This happens because changes of X_w and Y_w can be better perceived in the image plane.

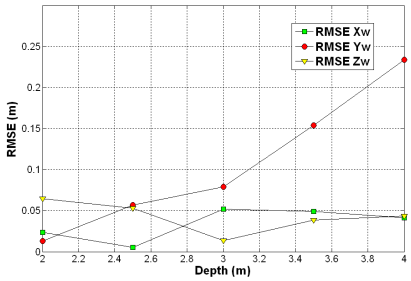

Fig. 6. Test results. MSE obtained by comparing the pose estimation results of the trinocular system with the known 3D positions at different depths.

The errors found in these experiments give an idea of the precision of the system, being $\approx 5cm$ for the X_w axis, $\approx 5cm$ for the Z_w axis, while for the Y_w axis, the precision depends on the depth. If a landing task has to be achieved, the adequate depth for this task is $\approx 3m$, whereupon a precision of $\approx 10cm$ will be the one obtained for the Y_w axis. The precisions that were obtained satisfy the requirements of the tasks we expect to accomplish (landing and positioning).

- **Flight tests**

The algorithm has been tested during different flights (the video sequences can be found in [25]). An interesting sequence to analyze corresponds to a landing task which has been performed in manual-mode (flown by the pilot). In Fig. 7 (figures a,b, and c), the normalized signals of the helicopter's state estimator and the vision system are compared regarding the UAV's position estimation. It is important to notice that, in spite of being a manual-mode flight (having thus strong movements), the algorithm follows the landmarks in the different frames and the reconstructed values are consistent with the real movements experimented by the helicopter. Analyzing the different signals, the vision system signals (red lines) are consistent with the position of the helicopter shown in the image sequences (the helicopter moves to the left and right, forward and backward, up and down of Camera 2). However, the UAV values (green lines) present different variations at the end of the landing phase that do not correspond with the helicopter's real movements, especially when the helicopter has landed (from frame 6750). Therefore, it is demonstrated that the vision system is not only capable of sensing the different movements adequately, but also that it is capable of sensing small variations in position ($\pm 50cm$), improving the position estimation in the different axes.

Another analysis has been done regarding the *yaw* (θ) values that are estimated when the helicopter has been rotated in different directions. In Fig. 8, it is possible to see the similar behavior of the signals, obtaining an RMSE of 4.3 considering the measure given by the helicopter's state estimator as ground truth.

- **Algorithm performance**

Table I presents the execution time for the different stages of the algorithm. The image processing stage (as expected)

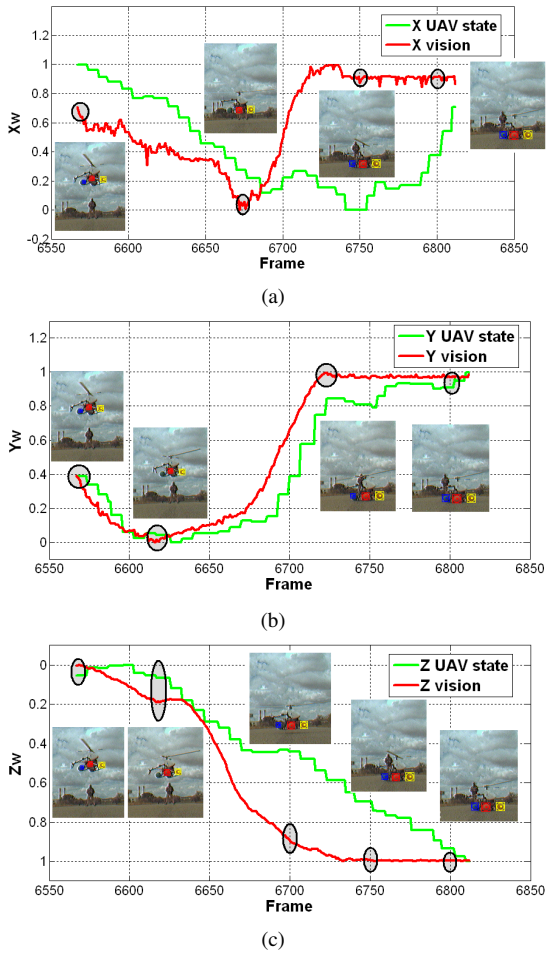

Fig. 7. Algorithm results vs helicopter's state estimator. The graphics correspond to a manual-mode landing task. The vision system's estimation is compared with the UAV's state estimator values and saved images, demonstrating that the vision system not only senses the different helicopter's movements adequately, but also that it is capable of sensing small variations in position ($\pm 50cm$).

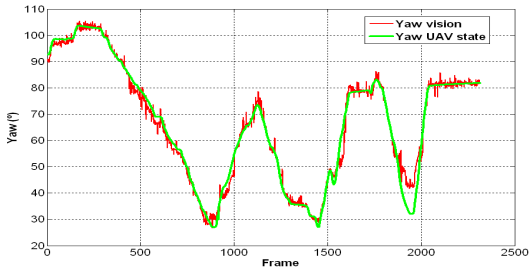

Fig. 8. Algorithm results vs helicopter's state estimator II.The signal of the *Yaw* angle of the helicopter state estimator (green line) is compared with the raw data of the vision-based estimation (red lines).

takes the largest CPU time. The average time for solving the pose estimation process is $\approx 56.7ms/frame$, allowing a frequency of $\approx 17.63frames/sec$, which proves that the algorithm can be used for real-time applications.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented and validated a real-time vision system that estimates the UAV's position and

TABLE I

COMPUTATIONAL COST

ALGORITHM STAGE	TIME (ms/frame)	PERCENTAGE
Capture	≈ 3.455	6.1%
Image Processing	≈ 51.9	91.5%
Triangulation	≈ 0.1	0.1%
Other tasks	≈ 1.3	2.3%
Total	≈ 56.7	100%

orientation based on information extracted by three cameras located on the ground. The algorithm has been tested in indoor and outdoor environments, the latter corresponding to real flight tests.

The experimental results show that the estimation is consistent regarding 3D real data with approximate errors of $\pm 5cm$ in X_w and Z_w axes, and of $\pm 10cm$ in the Y_w axis (at $3m$ depth), which satisfies the requirements of the tasks we expect to accomplish (landing, takeoff and navigation).

The comparison of the information provided by the vision system with that of the UAV's state estimator shows that the vision system improves the position estimation, especially the helicopter's height estimation, whose current accuracy based on GPS is $\pm 0.5m$ or lower when the vehicle approaches the ground. Additionally, it was possible to see that small variations in position are better perceived with the vision system, which makes this system suitable for maneuvers at low heights and close to structures where precise movements are required.

Our future work will focus on the implementation of a *Kalman Filter*, in order to smooth the estimation values. It will also aim to close the UAV's control loop with the extracted information in order to perform positioning and landing tasks. Additionally, we will study other techniques for feature extraction and matching employing other key points. All these new characteristics will be useful for the improvement of the UAV's pose estimation, which is crucial to allow UAVs to operate at low heights and to perform landing, take-off, and other ground-air cooperation tasks.

VII. ACKNOWLEDGMENTS

The work reported in this paper is the product of several research stages at the Computer Vision Group of the Universidad Politécnica de Madrid. The authors would like to thank Jorge León for supporting the flight trials and the I.A. Institute of the CSIC for collaborating in the flights consecution. We would also like to thank the Universidad Politécnica de Madrid, the Consejería de Educación de la Comunidad de Madrid and the Fondo Social Europeo (FSE) for the Authors' PhD Scholarships. This work has been sponsored by the Spanish Science and Technology Ministry under grant CICYT DPI 2007-66156.

REFERENCES

[1] Luis Mejias, Srikanth Saripalli, Gauvav Sukhatme, and Pascual Campoy. Detection and tracking of external features in a urban environment using an autonomous helicopter. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3983–3988, May 2005.

[2] Omead Amidi, Takeo Kanade, and K. Fujita. A visual odometer for autonomous helicopter flight. In *Proceedings of the Fifth International Conference on Intelligent Autonomous Systems (IAS-5)*, June 1998.

[3] S. Hrabar, G.S. Sukhatme, P. Corke, K. Usher, and J. Roberts. Combined optic-flow and stereo-based navigation of urban canyons for a uav. *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3309–3316, Aug. 2005.

[4] T.G. McGee, R. Sengupta, and K. Hedrick. Obstacle detection for small autonomous aircraft using sky segmentation. *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 4679–4684, April 2005.

[5] S. Saripalli, J.F. Montgomery, and G.S. Sukhatme. Visually guided landing of an unmanned aerial vehicle. *Robotics and Automation, IEEE Transactions on*, 19(3):371–380, June 2003.

[6] Cui Xu, Liankui Qiu, Ming Liu, Bin Kong, and Yunjian Ge. Stereo vision based relative pose and motion estimation for unmanned helicopter landing. *Information Acquisition, 2006 IEEE International Conference on*, pages 31–36, Aug. 2006.

[7] D. Hubbard, B. Morse, C. Theodore, M. Tischler, and T. McLain. Performance evaluation of vision-based navigation and landing on a rotorcraft unmanned aerial vehicle. *Applications of Computer Vision, 2007. WACV '07. IEEE Workshop on*, pages 5–5, Feb. 2007.

[8] Peter Corke. An inertial and visual sensing system for a small autonomous helicopter. *Journal of Robotic Systems*, 21(2):43–51, 2004.

[9] Alison Proctor Allen Wu, Eric Johnson. A vision-aided inertial navigation for flight control. *Journal of Aerospace Computing, Information, and Communication*, 2(9), 2005.

[10] Glenn P. Tournier, Mario Valenti, Jonathan P. How, and Eric Feron. Estimation and control of a quadrotor vehicle using monocular vision and moir patterns. In *In AIAA Guidance, Navigation and Control Conference*, pages 2006–6711. AIAA, 2006.

[11] Mario Fernando Montenegro Campos and Lúcio de Souza Coelho. Autonomous dirigible navigation using visual tracking and pose estimation. In *ICRA*, pages 2584–2589, 1999.

[12] Erdinç Altuğ, James P. Ostrowski, and Camillo J. Taylor. Control of a quadrotor helicopter using dual camera visual feedback. *Int. J. Rob. Res.*, 24(5):329–341, 2005.

[13] Gary R. Bradski. Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, (Q2), 1998.

[14] E. Trucco y A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.

[15] Pascual Campoy, Juan F. Correa, Ivan Mondragón, Carol Martínez, Miguel Olivares, Luis Mejías, and Jorge Artieda. Computer vision onboard uavs for civilian tasks. *J. Intell. Robotics Syst.*, 54(1-3):105–135, 2009.

[16] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000.

[17] Michael J. Swain and Dana H. Ballard. Color indexing. *Int. J. Comput. Vision*, 7(1):11–32, November 1991.

[18] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *Information Theory, IEEE Transactions on*, 21(1):32–40, Jan 1975.

[19] D. Comaniciu and P. Meer. Robust analysis of feature spaces: color image segmentation. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:750, 1997.

[20] Intel Corporation (2001). Open source computer vision library reference manual.

[21] Jean-Yves Bouguet. Camera calibration toolbox for matlab, 2006.

[22] Seth Hutchinson, Greg Hager, and Peter Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12:651–670, 1996.

[23] Luis Mejias, Srikanth Saripalli, Pascual Campoy, and Gaurav Sukhatme. Visual servoing approach for tracking features in urban areas using an autonomous helicopter. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2503–2508, Orlando, Florida, May 2006.

[24] Luis Mejias. *Control visual de un vehiculo aéreo autonomo usando detección y seguimiento de características en espacios exteriores.* PhD thesis, Escuela Técnica Superior de Ingenieros Industriales. Universidad Politécnica de Madrid, Spain, December 2006.

[25] Computer Vision Group. Universidad Politécnica de Madrid. COLIBRI project homepage. http://www.disam.upm.es/colibri, 2009.